**APYTHONPROGRAMTOIMPLEMENTKNNMODEL**

**Ex.No:9A Date of Experiment:18/10/24**

**AIM:-**

To implement a python program using a KNN Algorithm in a model.

# ALGORITHM:-

Step1: Import all the other necessary libraries(numpy as np, matplotlib.pyplot as plt and sklearn.tree,pandas as pd and seaborn as sns).

Step2: Select the number K of the neighbors.

Step3: Calculate the Euclidean distance of K number of neighbors of data points.

Step4: Take the K nearest neighbors as per the calculated Euclidean distance.

Step5: Among these k neighbors, count the number of the data points in each category.

Step6: Assign the new data points to that category for which the number of the neighbor is maximum.

Step7: Plot the graph “X” and “y” the values tested and predicted using seaborn.scatterplot() function.

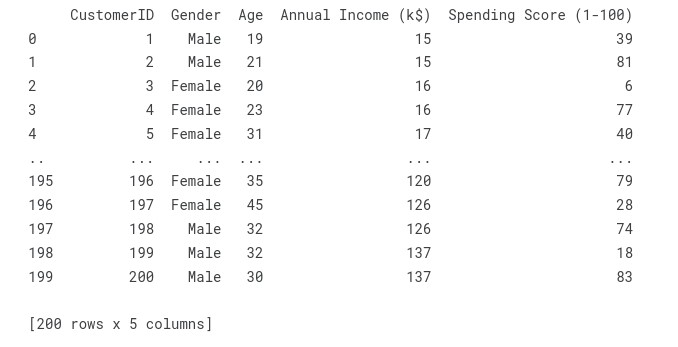
Step8: Print the confusion matrix of the model to know the accuracy of the model with support values for each class.

# IMPLEMENTATION:-

import numpy as np

import matplotlib.pyplot as plt import pandas as pd

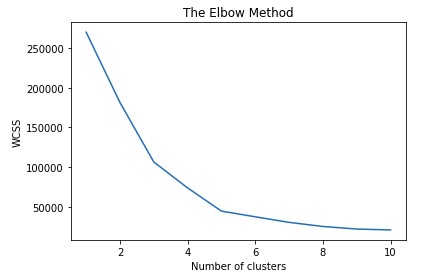
dataset = pd.read\_csv('../input/mall-customers/Mall\_Customers.csv') X = dataset.iloc[:,[3,4]].values print(dataset)



from sklearn.cluster import KMeans wcss =[] for i in range (1,11): kmeans = KMeans(n\_clusters = i, init = 'k-means++', max\_iter =300, n\_init = 10,

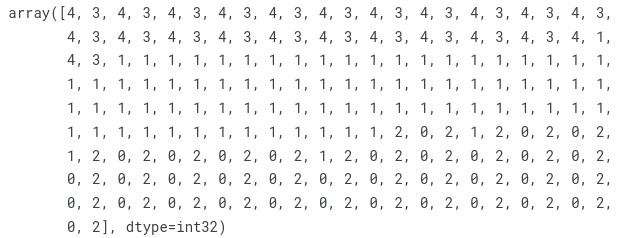
random\_state = 0) kmeans.fit(X) wcss.append(kmeans.inertia\_)

# Plot the graph to visualize the Elbow Method to find the optimal number of cluster plt.plot(range(1,11),wcss) plt.title('The Elbow Method') plt.xlabel('Number of clusters') plt.ylabel('WCSS') plt.show()

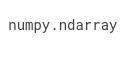


kmeans=KMeans(n\_clusters= 5, init = 'k-means++', max\_iter = 300, n\_init = 10, random\_state = 0)

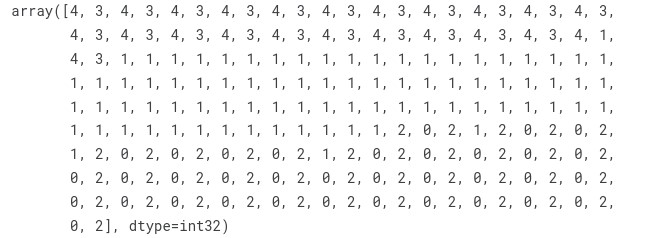
y\_kmeans = kmeans.fit\_predict(X) y\_kmeans



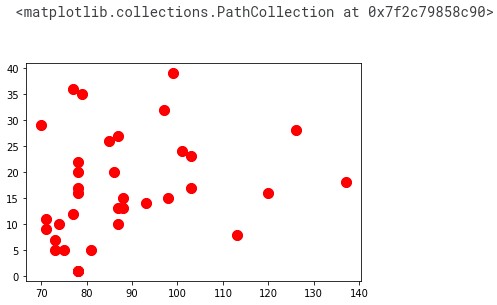
type(y\_kmeans)



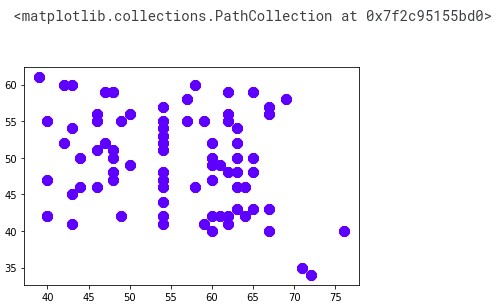
y\_kmeans



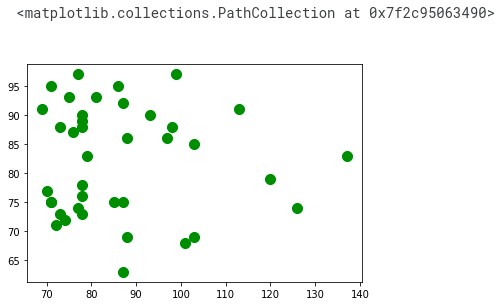
plt.scatter(X[y\_kmeans == 0, 0], X[y\_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')



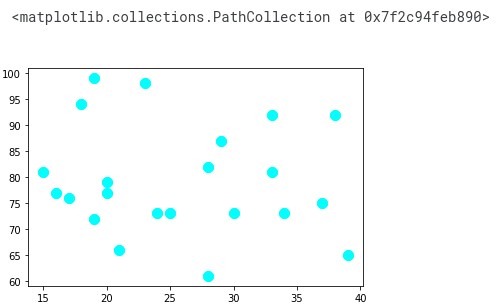
plt.scatter(X[y\_kmeans == 1, 0], X[y\_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')



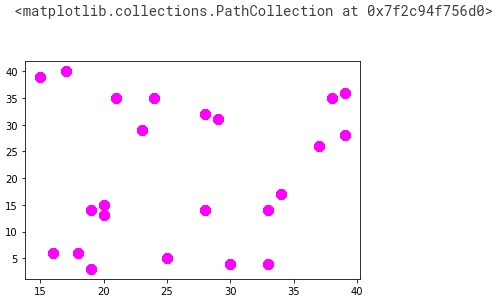
plt.scatter(X[y\_kmeans == 2, 0], X[y\_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3')



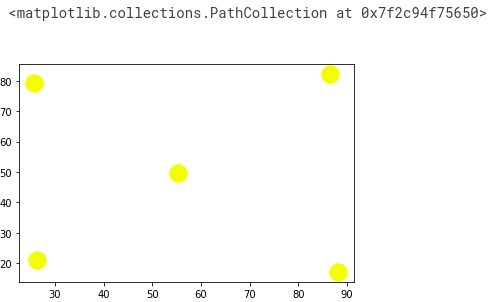
plt.scatter(X[y\_kmeans == 3, 0], X[y\_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')



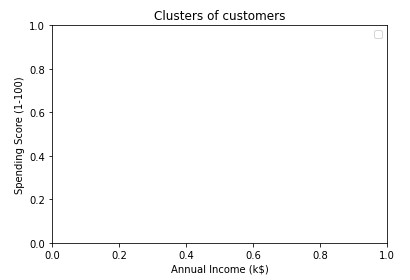
plt.scatter(X[y\_kmeans == 4, 0], X[y\_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')



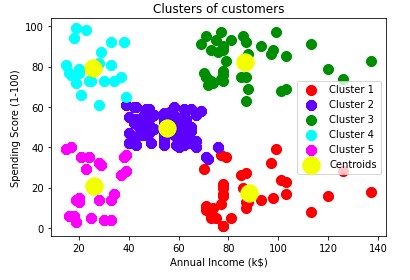
plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s = 300, c = 'yellow', label = 'Centroids')



plt.title('Clusters of customers') plt.xlabel('Annual Income (k$)') plt.ylabel('Spending Score (1-100)') plt.legend() plt.show()



plt.scatter(X[y\_kmeans == 0, 0], X[y\_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1') plt.scatter(X[y\_kmeans == 1, 0], X[y\_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2') plt.scatter(X[y\_kmeans == 2, 0], X[y\_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3') plt.scatter(X[y\_kmeans == 3, 0], X[y\_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4') plt.scatter(X[y\_kmeans == 4, 0], X[y\_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5') plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s = 300, c = 'yellow', label = 'Centroids') plt.title('Clusters of customers') plt.xlabel('Annual Income (k$)') plt.ylabel('Spending Score (1-100)') plt.legend() plt.show()



# RESULT:-

Thus the python program to implement KNN model has been successfully implemented and the results have been verified and analyzed.